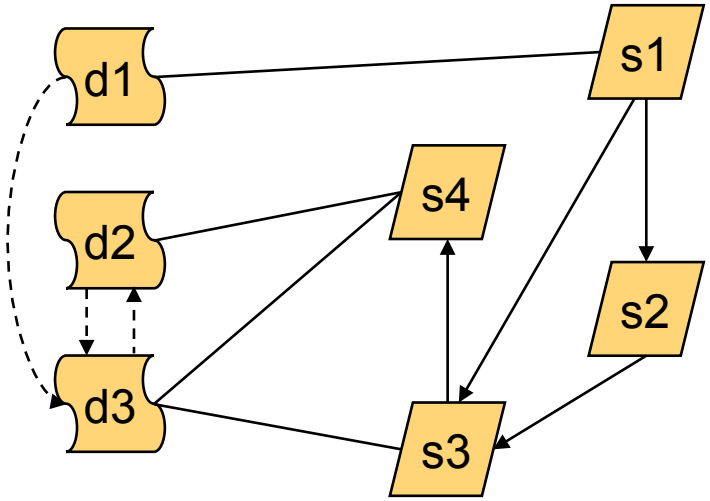
Combining Textual and Structural Analysis of Software Artifacts for Traceability Link Recovery

Collin McMillan, Denys Poshyvanyk, Meghan Revelle
May 18, 2009

College of William & Mary
Semeru Research Group

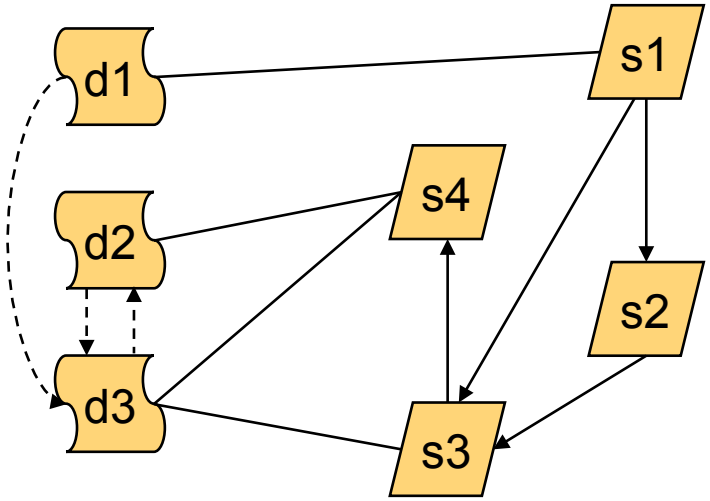
Motivation



- Software Artifacts
  - Source Code
  - Documentation
- Traceability Links
- Direct Recovery Methods
  - Textual comparison among documentation artifacts
  - Similar documentation may not be similar textually

- **Our goal:** Recover links among documentation elements using structural and textual artifacts; we call this *indirect link recovery*

Background



- Structural Analysis
  - JRipples[1] (static)
  - Program Dependency Graph
  - Provides source-source links
- Textual Analysis
  - Vector Space Model (VSM)
  - Latent Semantic Indexing (LSI)
  - Apply threshold or cutpoint to harvested similarities
  - Able to operate on documentation or text components of source code
  - Provides documentation-source links
  - Traditionally used for documentation-documentation links

[1] http://jripples.sourceforge.net/

Related Work

- Antoniol, G., Hayes, J. H., Gueheneuc, Y., and Di Penta, M., "Reuse or rewrite: Combining textual, static, and dynamic analyses to assess the cost of keeping a system up-to-date", in Proc. of IEEE ICSM'08, 2008, pp. 147-156.
- De Lucia, A., Fasano, F., Oliveto, R., and Tortora, G., "Recovering Traceability Links in Software Artefact Management Systems", *ACM TOSEM*, vol. 16, no. 4, 2007.
- Hayes, J. H., Dekhtyar, A., and Sundaram, S. K., "Advancing candidate link generation for requirements tracing: the study of methods", *IEEE TSE*, vol. 32, no. 1, January 2006, pp. 4-19.
- Marcus, A., Maletic, J. I., and Sergeyev, A., "Recovery of Traceability Links Between Software Documentation and Source Code", *IJSEKE*, vol. 15, no. 4, October 2005, pp. 811-836.
- Zhao, W., Zhang, L., Liu, Y., Sun, J., and Yang, F., "SNIAFL: Towards a Static Non-interactive Approach to Feature Location", *ACM TOSEM*, vol. 15, no. 2, 2006, pp. 195-226.

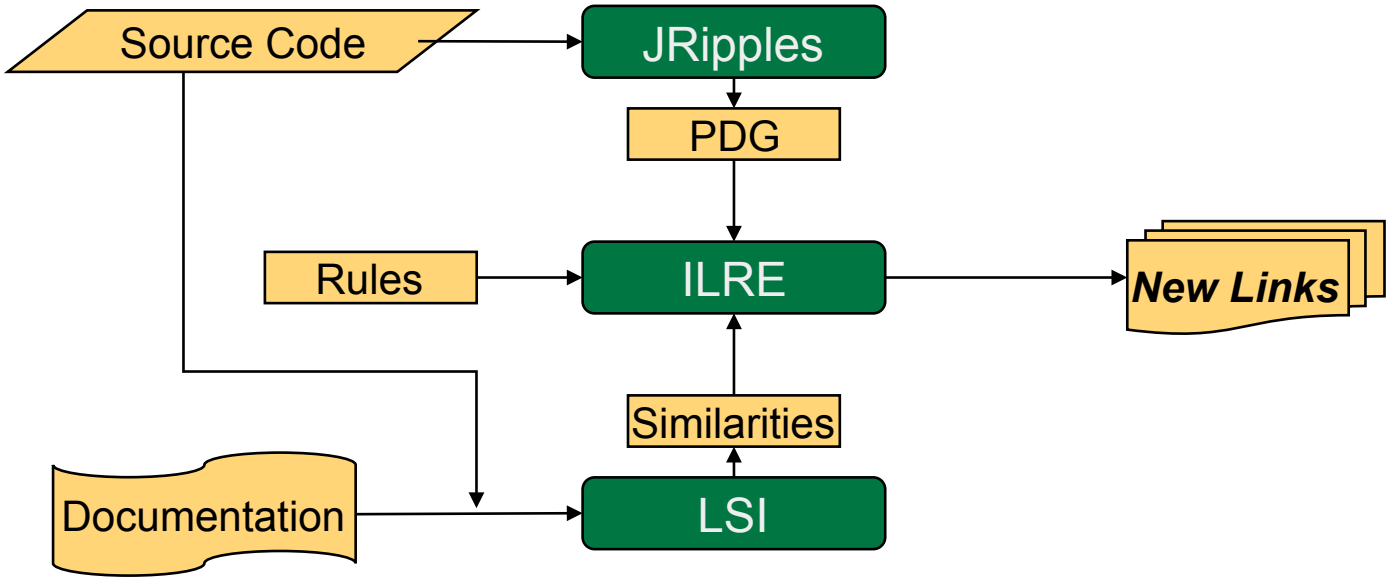
Example

- **Requirement A (R_A):** Waiting State. When the CoffeeMaker is not in use, it waits for user input. There are six different user input options: 1) add recipe, 2) delete a recipe, 3) edit a recipe, 4) add inventory, 5) check inventory, and 6) purchase beverage.

- **Requirement B (R_B):** Add Inventory. Inventory may be added to the machine at any time. The types of inventory in the Coffee Maker are coffee, milk, sugar, and chocolate. The inventory is measured in integer units. No inventory may be taken away from the CoffeeMaker except by purchasing a beverage. Upon completion, a status message is printed and the CoffeeMaker is returned to the waiting state.

- R_A and R_B have a text similarity of about 0.40 according to LSI

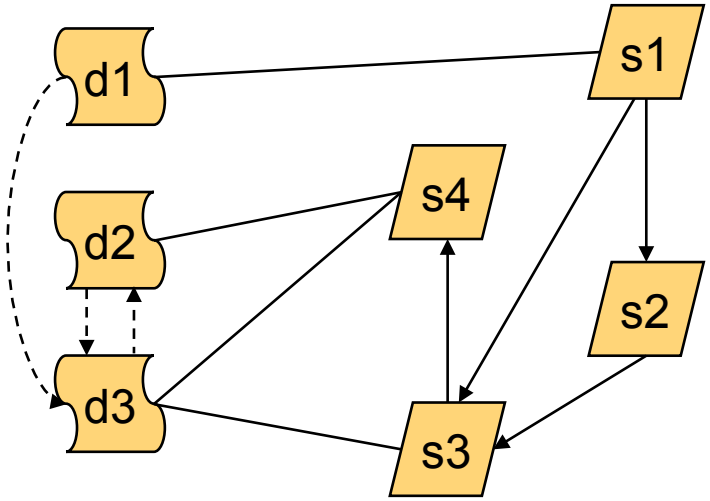
Approach



- Internals
  - All Java methods extracted as *source artifact nodes*.
  - Java method invocations added as *source-to-source edges*.
  - Requirements added as *documentation artifact nodes.*
  - *Documentation-to-source edges* added based on textual similarities.

Link Recovery

- Two Rules
  - Nodes d2 and d3 point to s4, so we deduce that a *document-to-document edge* (i.e., a link) should exist between those nodes because they share relevant source elements. This is the *same source rule*.



  - We also suggest a link between d1 and d3 because an edge exists from d1 to s1, from s1 to s3, and from d3 to s3. This is the *relative source rule*.

- The rules define how manipulations to the graph result in traceability link suggestions.

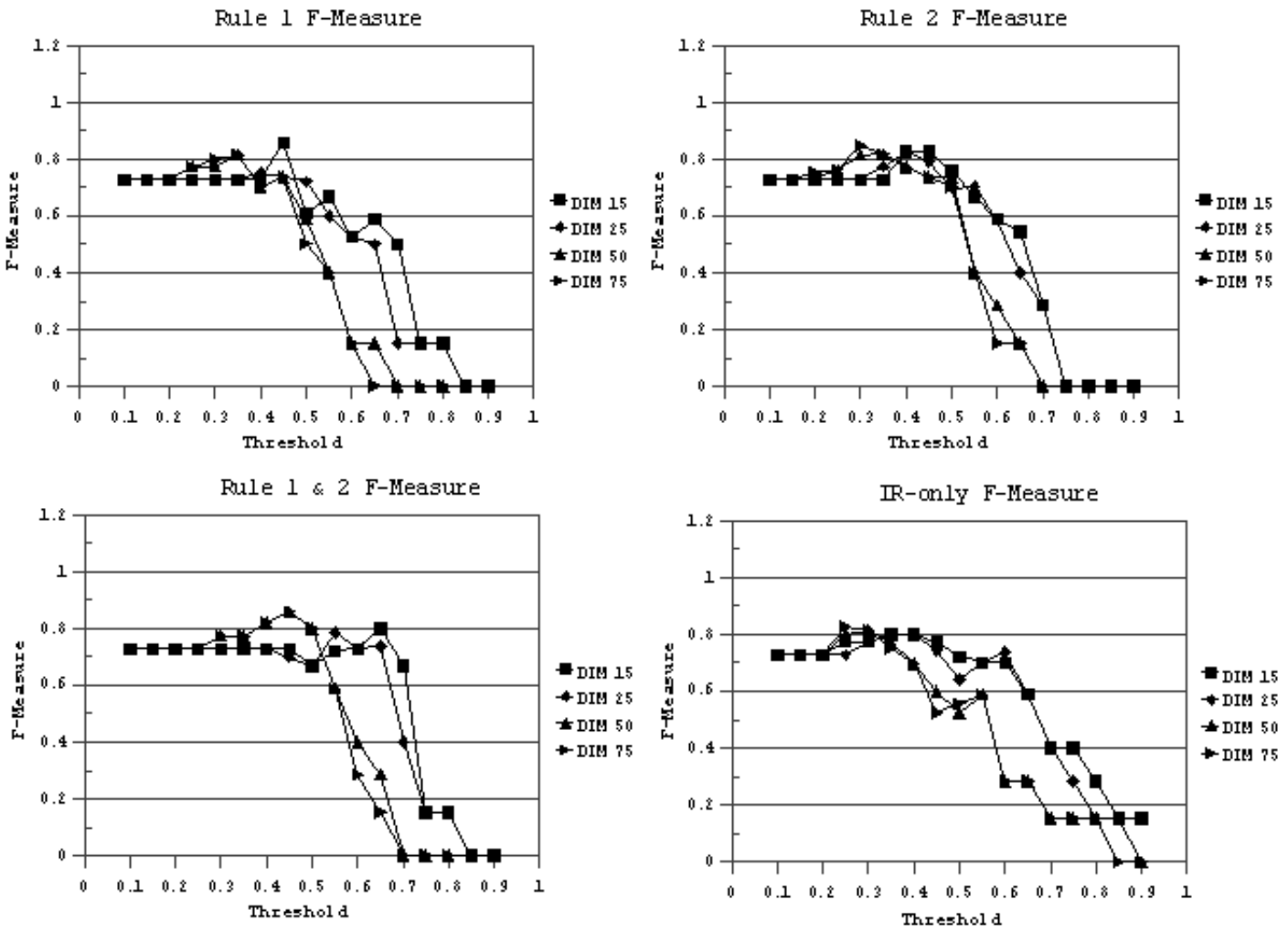
Case Study (1)

- CoffeeMaker[2]
  - ~1 KLOC, 136 Java methods
  - Implements intuitive functions of a coffee maker
  - Requirements documentation explicitly provided
- Oracle creation
  - For every pair of requirements A and B, we asked three graduate students: “Do you expect the implementation of A to overlap with that of B?”
  - Answers combined by voting

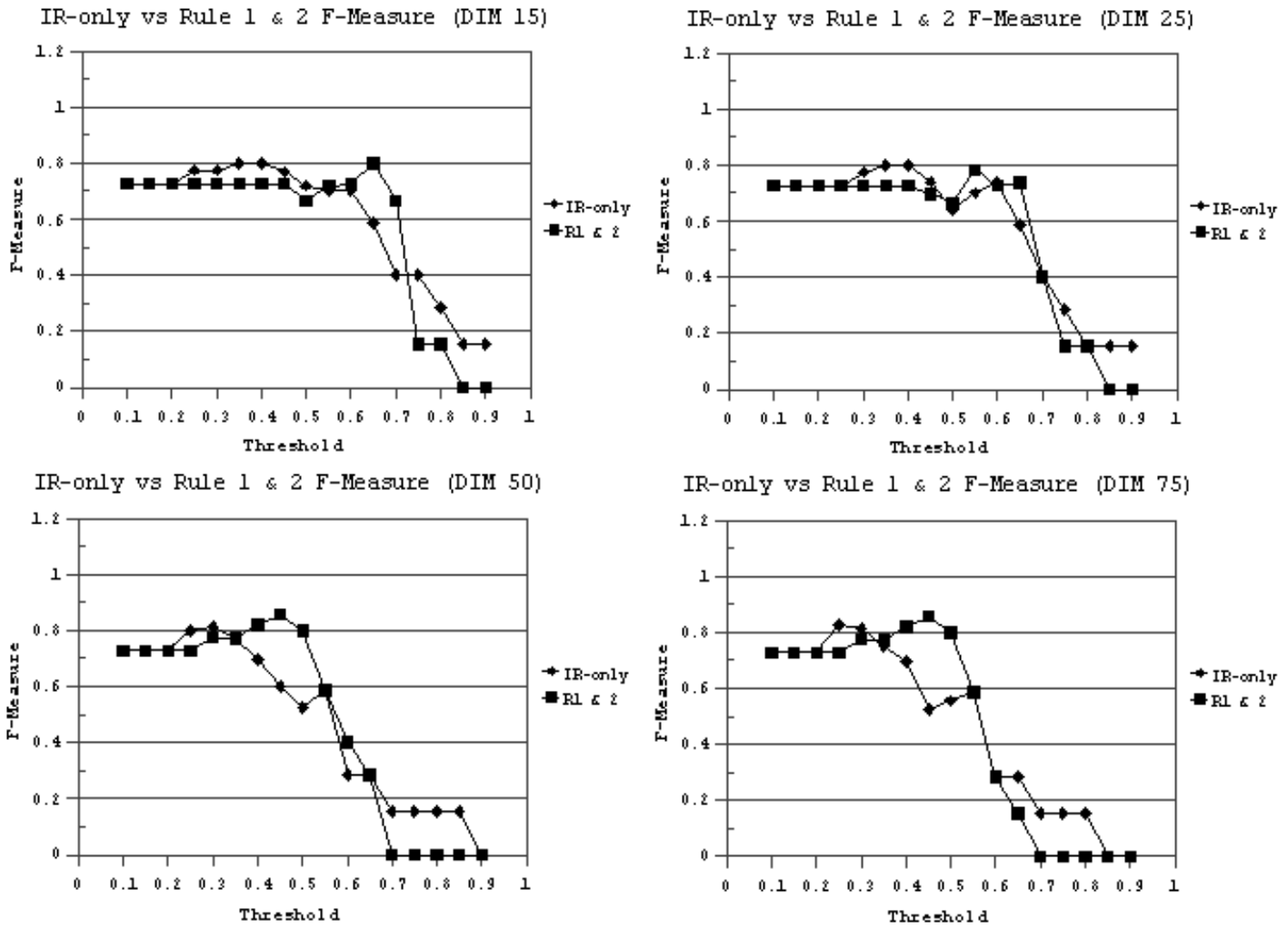
[2] http://agile.csc.ncsu.edu/rose/

Case Study (2)

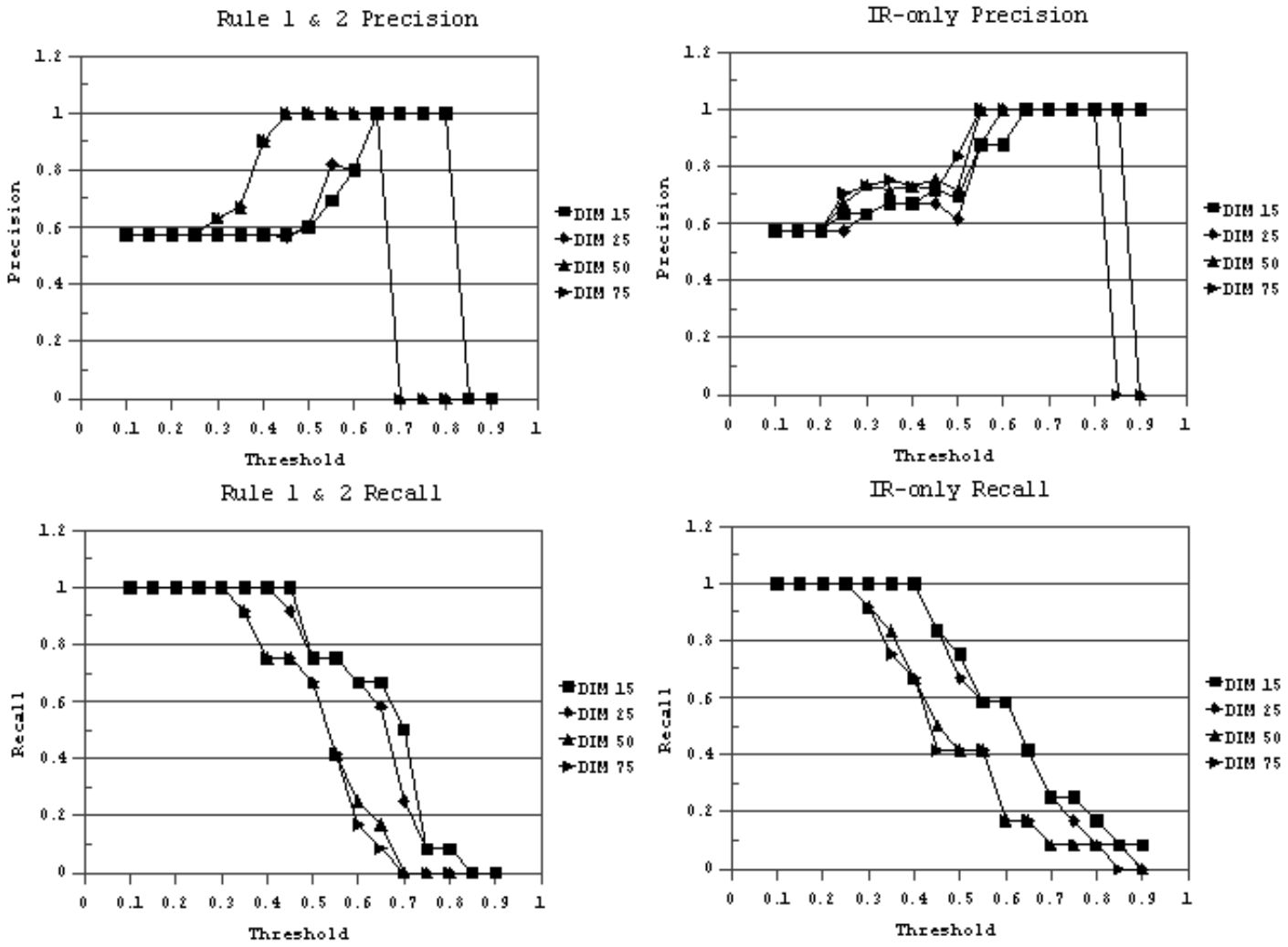
- Two independent variables studied
  - IR threshold for selecting documentation-source links
  - DIM used during SVD stage of LSI
- Metrics
  - *Precision* is the percent of suggested links that are correct.
  - *Recall* is the percent of correct links that are suggested.
  - The *f-measure* is the harmonic mean of precision and recall.
- IR-only (LSI) for comparison
  - Textual similarities of documentation elements from LSI
  - Threshold applied as for the IR section of our approach



- Do our rules perform better together or individually?



- How do our rules compare to the IR-only approach?



- How are precision and recall affected by the threshold?

Conclusions

- Existing software artifact analysis methods can be effectively combined for traceability link recovery.
- A software system's *structural* information can be harvested usefully to compare *text-based* artifacts.
- Need more software systems with existing benchmarks to better evaluate our approach.

- Semeru: http://www.cs.wm.edu/semeru/